A PROTOTYPE RING-STRUCTURED COMPUTER NETWORK USING MICRO-COMPUTERS

Keith Albert Hirt

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A PROTOTYPE RING-STRUCTURED

COMPUTER NETWORK USING MICRO-COMPUTERS

by

Keith Albert Hirt

Thesis Advisor: R. H. Brubaker, Jr.

December 1973

Approved for public release; distribution unlimited.

A Prototype Ring-Structured Computer Network Using Micro-Computers

by

Keith Albert Hirt
Lieutenant, United States Navy
B. S., Purdue University, 1969

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the
NAVAL POSTGRADUATE SCHOOL
December 1973

ABSTRACT

Architecture of a Ring communication network based on the Distributed Computing System (DCS) at the University of California, Irvine is presented. Control of the network is distributed in time among the active processors for reliability. A pseudo-bipolar violation protocol, modeled after the Farmer and Newhall ring at Bell Labs, is used to implement synchronization, control passing, and variable length messages. The prototype system is designed to make extensive use of micro-computers for flexibility in design. Ring organization and protocol are discussed with the major emphasis on the use of an INTEL MCS-8 Micro-Computer at the interface of a teletype to the ring. The MCS-8 program is written in PL/M-a higher level language for INTEL's micro-computers.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENT

The ring network project is under the direction of Assistant Professor Raymond H. Brubaker, Jr.

The author wishes to express his thanks to Assistant Professor Brubaker for his insights into bit-level hardware and to Professor David Farber at UC, Irvine, for the informative briefing and tour of the DCS System. Special thanks are in order for my wife, Janice, whose patience and understanding helped see this thesis through to completion.

I. INTRODUCTION

A project is underway at the Naval Postgraduate School to establish a local ring-structured communication network. This network will link present and future computers on the NPS grounds to provide a more efficient utilization of computing resources. This paper is a first report on the progress of that project.

Investigation of several existing computer networks led to the selection of the Distributed Computing System (DCS) developed at the University of California at Irvine as a model. The main features of the DCS that made it suitable as a model are reliability, relatively low cost, and its processor-independent communication's protocol. The ring structure of the DCS was retained along with the distributed control philosophy. The main differences between the proposed system and DCS are (1) the use of micro-computers at the ring interface (RI) to replace as many RI hardware functions as possible with software, and (2) the bipolar violation concept for start-of-message (SOM) and end-of-message (EOM) indication, synchronization (SYN), and control (CTL) passing. These violations will be referred to as "tokens."

System topology and protocol are discussed and the use of a micro-computer in the prototype system is presented. This thesis concentrates on the use of an INTEL MCS-8 micro-processor at the interface of a host to the communication ring. The micro-computer acts as a buffer between a host and the Ring Interface. It can control host communication with the ring and free the host processor from most of the "handshaking" required for communication with other processors. The term "smart terminal" has been used to describe the role of the micro-computer. A specific example incorporating a teletype (ASR-33) as the host is

programmed in PL/M and implemented on the MCS-8.

Micro-computers were chosen for 3 reasons:

1. Availability. The school has several micro-computers at its disposal (INTEL MCS-4 and MCS-8).

2. Flexibility. Micro-computers provide flexibility in design not found in a hardware-only configuration at the ring interfaces. Modifications to host/Ring Interface protocol can be accomplished by altering micro-processor software instead of changing hardware design.

3. High Level Programming Language. PL/M for INTEL's 8-bit micro-computers is supported by a compiler written in ANSI standard FORTRAN IV. It is available on the IBM-360/67 and XDS-9300 computers at tne Naval Postrgaduate School. PL/M is designed for the systems programmer and retains the efficiency and control of an assembly level language.

II. SYSTEM TOPOLOGY

A. REQUIREMENTS

The main requirement of the system is to provide a reliable, low cost communication link between the computer systems presently located at NPS and to be flexible enough to add future computer systems to this network without having to modify network protocol (and therefore RI design). Reliability was also considered important from the standpoint of possible applications of this communication structure to the operational Navy.

1. Reliability

Reliability is achieved by distributing control of the ring amoung the active users with a fail-soft philosophy of control. No single processor has the responsibility for the control of the ring. Control is distributed in time over all the processors in the network. The failure of any processor will not hinder communication between the remaining processors.

The ring is never idle for more then short periods at a time. If processes are not in direct communication, the ring idles by relaying control (CTL) tokens between active nodes. Maintaining the ring in a semiactive state provides a positive control mechanism that minimizes the time required to recognize control failures. At system light-off, the first node to be activated will commence broadcasting violation tokens. This is strictly a function of the Ring Interface and is independent of host status. A node with a message for transmission will take control of the ring by replacing the control token (CTL) with a start-of-message (SOM) token. After message transmission is

complete, the transmitting node will send a CTL token and wait until it either recognizes an SOM from another node or its host has another message to transmit. If the node that is currently transmitting (either a message or token) should for some reason fail, the remaining nodes will go into a "time-out" state. An interface completing its time-out without receiving an SYN or CTL violation token will assume control of the ring. In effect each active node monitors the control of the ring. Any node can take control if necessary to insure control reliability.

Each interface has a unique delay time with enough separation in times to prevent two from both simultaneously attempting to transmit tokens. An RI that times-out, transmits an SYN token, and fails to receive its transmitted token, will signal its host. The host can then make the decision on whether to exit from the ring. An alternate procedure could be to switch to a standby ring and attempt to join this ring in a normal manner.

2. Response/Bandwidth Requirements

A range of devices from low speed to high speed must be accomodated. The limiting factor in the system is the slowest rate at which messages can be placed onto or taken off the ring. In the prototype system the limiting component is the MCS-8 and its relatively long instruction cycle time (12.5 microseconds). Specifically, to make successive 8-bit bytes available to the Ring Interface for transmission, the program requires 59 machine cycles. This limits ring speed to approximately 10Kb (Kilobits per second). Even at this relatively slow speed, a number of printers and card readers could be driven. A remote batch entry system for the IBM-360 will be attempted in the near future using the NPS Ring. 10Kb is considered adequate for such an attempt.

The ring speed could be increased significantly by

allowing the RI direct access to the MCS-8's random access memory. The RAM ready line makes this approach feasible, but then the role of the micro-computer would be reduced and the hardware complexity of the Ring Interface would be increased. It is planned that later ring interfaces based on micro-programmed MOS and TTL logic will allow ring speeds on the order of 1000Kb. This speed would require buffers between slow speed hosts (like the MCS-8/TTY) and the Ring Interface.

3. Flexibility

The ring is designed to move bits. It is not dependent on a particular machine's method of representing characters or data. Communication is independent of processor word length and therefore can support a variety of processors and components. Communication between two nodes is of course dependent on the particular processor's representation of information. But as far as the ring is concerned, the transmission is just 0's and 1's.

Flexibility is also achieved with the use of micro-computers at the Ring Interface with the host processor. The prototype system uses a teletype and MCS-8 to output messages to the ring for ring testing and performance evaluation. This paper also provides MCS-8 software support for file creation and modification from a TTY via the ring to another processor such as the XDS-9300. The software system is flexible enough to allow editing and execution of the file and dumping of the output back to the TTY.

4. Simplicity and Cost

To keep cost down and maintain a high degree of simplicity, off-the-shelf components are used to the maximum extent possible. The ring itself is shielded,twisted pair wire. The Ring Interface is the most complex component from

the standpoint of construction and design. The TTY connects to the MCS-8 using a Universal Asynchronous Receiver-Transmitter for serial to parallel and parallel to serial conversion. The MCS-8 connects to the Ring Interface using two output and two input ports.

B. ORGANIZATION

The proposed network architecture is the data ring or loop system. This type of system connects all processors and/or terminals by a common uni-directional bus. Farmer and Newhall [4] at Bell Labs developed a ring designed to handle bursty traffic using a bipolar violation technique. Control is provided for by a control computer whose function it is to monitor and regulate traffic on the ring. The introduction of a network controller inserts a critical component into the system. If this component fails, the ring cannot function. David Farber at the University of California, Irvine, [1,2,3] overcame this handicap by distributing control amoung the various processors currently active in the network. This DCS philosophy was adapted for NPS ring organization.

One feature of the Farmer and Newhall system was incorporated into the NPS Ring. Pseudo bipolar violations are used for SOM and EOM indication, SYN, and CTL tokens. The Farmer and Newhall ring uses a broadband system based on digital PCM. The transmission scheme chosen for the NPS Ring is digital. Bipolar signaling is simulated by using logic '1' to logic '0' and logic '0' to logic '1' transitions.

The DCS uses a text bit count as part of the message to enable Ring Interfaces to determine when text leaves off and error checking bits begin. This flexibility enables variable length messages to be sent over the ring. One problem results from this procedure. If an error should occur in the bit field used to indicate text length, synchronization would be lost on the ring. The pseudo

violation used at NPS consists of sequences of physical combinations of one's and zeros's that cannot appear in text and are unlikely in error sequences. The type of signalling is digital with data bits (text) requiring two physical bits on the ring. The Ring Interface adds redundant bits to processor data to form the pseudo bipolar effect. A processor '0' and '1' are encoded as a '01' and '10' respectively. This procedure leaves a '11' and a '00' for ring violations. Using the '111' violation as part of the SOM and EOM tokens gives the NPS Ring variable length message capability.

This method is also considered advantageous for error detection. The probability that a '01' sequence on the ring (data '0') would be changed into a '10' (data '1') is small. It would be more probable that a data bit would be changed into a violation which would be picked up by the RI's as an error. Additional error detection is provided by cyclic redundancy check bits added to message text by the RI. The pseudo bipolar transmission scheme may make these check bits unnecessary. If the great majority of errors produce violations that can be detected by the RI, the CRC check can be removed. This will be proved or disproved by observing actual error rates on the ring. Initial RI design will provide a means of determining how many errors are detected by the CRC that did not produce violations.

1. Ring Structure

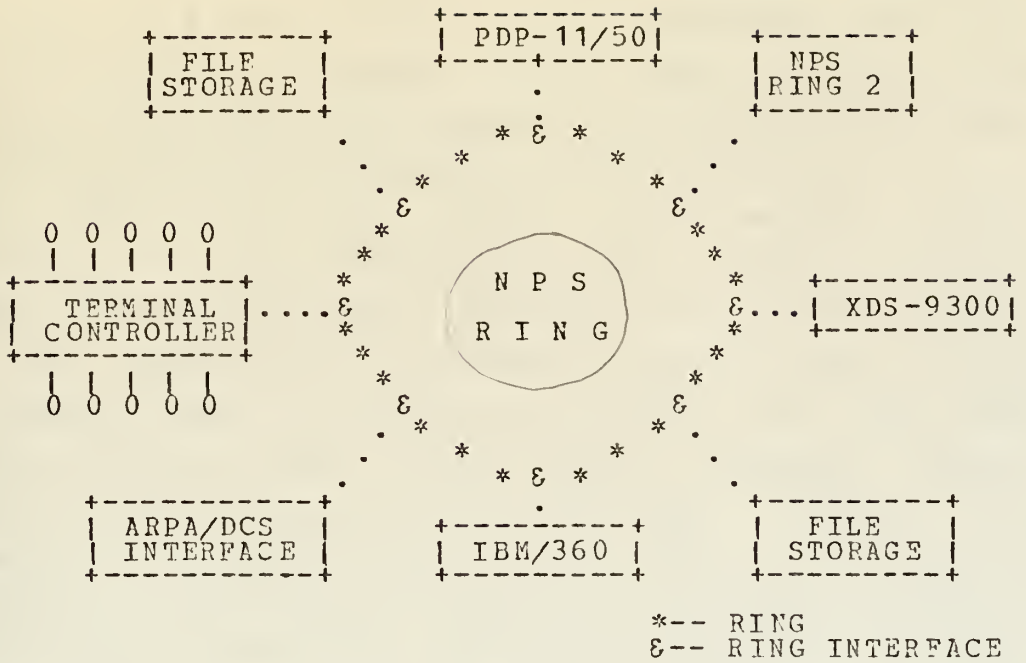


Fig.1. Possible NPS Ring

The proposed system is of the type shown in figure 1. The ring is uni-directional with one node (consisting of a host processor and its Ring Interface) in a control state. An active node is defined as a host processor with a process (at least one) capable of using the ring as a communication's medium. The DCS feature of having processes rather than processors as the logical communications entity has been retained. A processor may have more than one process capable of ring communication. The respective RI is "loaded" with the processes current at a particular host during the start-up phase. Loading of process names is accomplished by providing an address to the Process Name Memory (PNM) in the Ring Interface. The bit determined by this address is set to a '1' to indicate an active process name. Active process names are deleted by resetting this

bit.

The node in a control (master) state will either be transmitting a message to another process or relaying CTL violation tokens to maintain the ring active. Transmission of tokens is a function of the Ring Interface and not the host or a host process. Active nodes will stop relaying control tokens when they receive an SOM from another node or when their host has a message for transmission.

Each node must examine each message header to determine if the destination code matches a previously loaded host process name. Each node forwards the message to the next node even if a destination match occurs. Messages are removed from the ring by the source node and replaced with violation codes. This insures messages complete a circuit of the ring once and only once. Receipt of the message is signified by use of match and accept bits appended to the message. These bits are made available to the sending process and a determination is then made by the originator (in the host) as to whether the message is to be re-transmitted.

2. Ring Interface

The purpose of the Ring Interface is to 1) reshape and forward signals on the ring, 2) recognize control tokens, 3) compare destination process names with processes active on its host, 4) signal host and assemble bits of the incoming message into 8-bit bytes and pass them to the host, 5) determine whether overrun has occured, 6) check CRC bits for an error in the received message, 7) set the match, accept, and bad CRC bits accordingly, 8) replace the CTL violation with an SOM when its host process has a message ready for transmission, 9) switch outgoing messages onto the ring, 10) calculate and insert CRC bits onto the ring after generating an EOM violation after text transmission, 11) output a '0' match, a '0' accept, and a '0' bad CRC bit for

resetting by the destination process' interface, 12) examine returning message status bits and make them available to the host for examination, and 13) maintain the ring active with SYN or CTL violations when in a master status. In addition the RI plays a significant role in ring synchronization and error recovery.

This thesis will not dwell on the hardware structure of the Ring Interface, but will treat it functionally as a "black box".

3. Role of the Micro-Computer

As mentioned previously, one purpose of the NPS ring organization was to replace hardware with software where possible for maximum flexibility in design. It is anticipated that once the Ring Interface functions are firm and construction is complete, the RI will be processor independent. The RI for the IBM/360 will be the same as the RI for the PDP/11. Processor differences will be "absorbed" by use of micro-processors (or custom logic interfaces later, for higher speed).

In the prototype system, the MCS-8 gives a TTY a process status. It provides a means of identifying the TTY as a process. It allows initializing and changing message destination bits. Characters are buffered in MCS-8 RAM a line at a time. Signalling of the RI to transmit a message and making the message available for transmission are under the control of MCS-8 software. On the receive side the MCS-8 recognizes notification that the RI has a message for the TTY. After storing the message in RAM, overrun and CRC errors are checked prior to dumping the message to the TTY. In the current system only one incoming message (78 characters or less) is buffered in RAM. Further messages are blocked by disabling the receiver section of the RI until the message in RAM has been transfered to the teletype. In this case the RI would set the match bit and

leave the accept bit '0'.

4. Host Processors

The ring should support a variety of host processors. The protocol is flexible enough to enable dissimilar devices to utilize the ring. A host is any component capable of either transmitting and/or receiving messages via the ring. An example of a transmit-only host would be a system status message broadcaster. A node desiring system status reports would load the broadcaster's process name into its Ring Interface RAM. Periodic status reports could be broadcast over the ring and received by only those units desiring status reports. A receive only example would be a line printer or a card punch.

The RI could be simplified somewhat for the "one-way" host. There would be no need for RAM in the RI for process name loading. A receive-only or transmit-only node could have its name hardwired in interface logic. A receive-only host would not require circuitry for switching messages onto the ring, but would still have to decode CRC bits and alter message status bits. A transmit-only node would not have any use for a destination process name decoder. If transmitted messages are periodic, the node could even do without CRC/match/accept bit monitors.

In the general "two-way" host a nucleus of some sort would be required. The nucleus would be required to make decisions on system status and take appropriate action. It would have to recognize error conditions and initiate diagnostic routines for failure localization. The nucleus should be as complex as ring reliability requirements dictate. If a secondary ring is provided, the nucleus could selectively order participating nodes to shift to the secondary ring as part of a fault isolation procedure.

The same basic nucleus would reside on several of the major processors. The sophistication of the program

might preclude implementation of the nucleus program on a micro-processor.

III. PROTOCOL

The main objective of the NPS Ring protocol is to permit process communications between heterogeneous components such that it is possible for each component to communicate without knowing the physical placement of the other processors. The protocol should be processor independent and support variable length messages regardless of code structure used in the text of the message.

The prototype system transmitted message consists of the following fields:

1. Start-of-Message token (violation).
2. Destination process name.
3. Source process name.
4. Text.
5. Sequence field.
6. End-of-Message token (violation).
7. Cyclic Redundancy Check bits.
8. Match bit.
9. Accept bit.
10. Bad CRC bit.

A. DISTRIBUTED CONTROL

1. Control Token (CTL)

The assumption of control of the ring is accomplished by replacing a control token with a start-of-message violation. The control token is a '11101010' physical sequence on the ring. The '11' header in the token is a violation code since it cannot appear in text. There is a 9 bit (4 and 1/2 data bit) delay at each active node. The delay is necessary to recognize and replace the token. Having the bits available at the RI

simplifies the logic required to make the decision on whether a token has been received. Changing from a CTL sequence ('11101010') to an SOM sequence ('11100011') becomes just a matter of switching an SOM suffix in place of the CTL suffix held in the delay shift register.

An alternate RI design could remove this delay. When the '1110' prefix is received the RI could shift the '0011' suffix onto the ring and look for a difference between what was received and what it transmitted. If no difference is detected then a control token was not received and the node assumes the SOM belongs to another node. The node then returns to the relay state and decodes the message as if it was from another source. If a change is detected in the suffix, the node assumes it has taken over control of the ring and switches a message onto the ring. This approach would require more complex timing and control logic at the RI than the method selected for the NPS Ring. In this case the receiver section of the RI about to assume control must be phase-locked with its transmitter since bits are being modified as received.

b. Synchronization

A major problem in the distributed control approach to ring communications is the synchronization of active nodes. This is a three part problem. The first part is initial synchronization. Next is maintaining synchronization and loss-of-synchronization (LOS) recovery. The third is re-synchronization of the ring when a node either leaves or joins the active ring.

Under steady state ring conditions, the following protocol is observed. The master node will broadcast a CTL token and start a timer waiting for its receiver to detect the token. If the CTL token is not received prior to expiration of the timer, the RI will transmit an SYN token. This SYN time-out is different for each RI with enough

need [illegible] of this concept

separation to insure only one will expire during any one ring cycle. A ring cycle is defined as the maximum possible ring delay in bits, divided by ring speed in bps. This would be about .23 seconds at 10Kb ring speed and 256 nodes on the ring (9 bits delay per node). On receipt of an SYN, CTL, or SOM token, each node resets its SYN timer. When a node's SYN timer expires, it transmits an SYN and assumes control of the ring. The main purpose of the SYN and SYN time-out is to insure control of the ring is maintained. If the master node receives its SYN prior to expiration of the timer, after its SYN T/O again expires, it transmits a CTL token. This insures only one CTL can exist on the ring at the same time.

When a CTL is replaced by an SOM, control of the ring shifts to the node making the replacement. After receipt of an SOM, SYN time-out will not occur until the ring goes idle. The transmitting node removes its own message from the ring. When it receives the match, accept, and bad CRC bits, it transmits a CTL token and shifts to the relay mode. In the relay mode only valid CTL, SYN, and SOM tokens are passed to the next node. This insures noise does not get continuously propagated around the ring. If a token is altered by noise, it will be immediately removed from the ring, since the next node seeing it will not recognize the sequence as a token and will not relay it. Synchronization will be regained when an SYN timer expires and a node transmits an SYN token. When an SOM is received, the RI transitions to the "receive" mode and continues to forward the message to the next node. If an SOM is altered by noise, the message will be removed from the ring.

Initial ring synchronization occurs as above by active nodes initiating an SYN T/O and waiting. The first to have its time-out expire will transmit an SYN ('11100010'). When the SYN is received by the other nodes, they reset their timers. It is still possible for two SYN's to exist on the ring at the same time. Since each node uses its

unique time to initiate the first CTL token, only one CTL can exist at any given time and synchronization is guaranteed. This prevents an "after you"/"no, after you" condition from occuring.

Part two of the synchronization problem is maintaining synchronization. With no message transmission (just SYN or CTL violation tokens), synchronization occurs with each violation. During message passing synchronization occurs on SOM and CTL violations. Loss of bit synchronization would result in seeing violations in text. For example, the data sequence '100111' would be encoded as '100101101010' on the ring. Being one bit out of synchronization would produce at least two pseudo-bipolar violations, but no violation token would result. The selection of the SYN, SOM, EOM, and CTL violations as eight-bit strings with a '111' sequence for a prefix insures that these codes cannot be duplicated by synchronization errors occuring in message text. Violations ('00' or '11') could occur due to single bit errors in text. Therefore, these violations cannot be used for synchronization. In either case if a violation occurs in text, the CRC check is assumed to fail and the RI looks for a legal violation code on which to re-synchronize.

The third case of the synchronization problem occurs when a node leaves or joins the ring. A node may leave due to a normal termination or some failure (or on command of a fault isolation routine). Termination removes the 9 bits of delay at that node. The bits in the node's delay shift register are lost. If termination occurs in text at a data bit boundary, ring synchronization would not be affected. The CRC check would fail and the message would be re-transmitted. If the delay occurs between a data bit boundary a '11' or '00' violation would be produced which would be detected by the RI's. If the delay is removed while part of a token was in the register, an LOS condition results. This situation is handled the same as when a node

joins the ring and introduces an extra delay between token boundaries. Nodes join the ring in the relay mode and if the ring is idle, the incoming node will start its SYN time-out and ring synchronization would not be affected. If a token is altered by the introduction of the new node's delay, the invalid bit string would be removed from the ring by the next node in sequence and an SYN timer expiration will re-synchronize the ring.

B. RING INTERFACE/MICRO-COMPUTER PROTOCOL

Fig.2. Prototype System

The MCS-8 system used in the prototype (Figure 2) is built up from INTELLEC-8 modules. Communication with the Ring Interface is accomplished using two input and two output ports. One input and one output port are used for message transfer with the two remaining ports used for control and status information.

1. Message Transmission

When the MCS-8 has a complete message in RAM, it loads the data output port with an 8-bit destination code and signals the RI via the XMTR ENABLE line. The RI waits for a control token and replaces it with a start-of-message token. The RI then latches the first byte of information (the destination code) and notifies the MCS-8 that it has the byte using the XMTR READY line. The BYTE AVAIL output line is used by the MCS-8 both to indicate that the RI has

received the last byte of a message and to notify the RI that the next byte is available. The Ring Interface uses this line to make the decision on whether an "overrun" error has occured. An overrun condition exists on transmission if the micro-computer (or other host) cannot make bytes available fast enough for the RI to maintain message continuity on the ring. This error should not occur under normal conditions, since the speed of the ring must be limited to the data rate of the slowest host.

When the complete text has been made available to the RI, the MCS-8 loads a serial bit into the output port and signals "serial bit available" (message complete) by resetting the XMTR ENABLE line. After transmitting the serial bit, the RI switches CRC bits onto the ring followed by match bit, accept bit, and bad CRC bit. On return of these status bits, the RI makes them available to the MCS-8 and transmits a CTL token. On the basis of the message status bits, the MCS-8 can order re-transmission of the message or make its RAM buffer available for the next message from its host. On an accept-fail condition the MCS-8 will attempt to transmit the message a second time. If that attempt fails, a transmit-error message is typed at the TTY.

2. Receiver Protocol

When a Ring Interface detects an SOM, the RI interprets the next 8 bits as the destination process name. A match is then attempted between the destination code and the active process names on the RI's host. If a match occurs the RI makes the destination byte available to the MCS-8 and signals start-of-message using the SOM line. On detection of the SOM by the micro-computer, the destination code is stored in RAM and acknowledged using the BYTE RECEIVED line (the same line that was used for BYTE AVAIL during transmit operations). The RI then resets the SOM line to acknowledge storing of the last byte. The MCS-8 then

waits for another logic '1' signal on the SOM line indicating that the next byte is available. The RI can again make determination on the occurance of overrun errors.

After the EOM token is detected by the RI, it will signal end-of-message to the micro-computer, using the EOM line, and set OVERRUN and CRC ERROR lines accordingly. The RI also will set the match, accept, and bad CRC bits based on the detected errors. The micro-computer can then decide whether to ignore or transfer the message to its host.

C. MICRO-COMPUTER/HOST PROTOCOL

The host in the Prototype System is an ASR-33 Teletype using an MCS-8 to attain processor status. The intention was to provide a minimal host that would have its entire nucleus implemented on a micro-processor. The minimum capabilities would have to include message origination and a receive capability.

The teletype can be used to (1) load source process names into Ring Interface RAM, (2) load a destination name into MCS-8 RAM, (3) transfer up to 78 characters as message text to MCS-8 RAM, and (4) order transmission of a message. As a receiver the TTY can accept messages in USASCII code of up to 78 8-bit characters in length.

1. Input and Output

As mentioned previously, serial to parallel and parallel to serial conversion is handled using an Universial Asynchronous Receiver-Transmitter (UAR-T). Characters from the TTY are made available to one of the MCS-8's input ports (via the UAR-T). The UAR-T signals character availability using the Data Available (DA) line. When using the INTELLEC-8 I/O Module, character acknowledgement is automatic upon execution of the input instruction on that port. When the UAR-T is used alone, the Reset Data

Available (RDA) line must be strobed to permit acceptance of the next character from the TTY. The UAP-T also provides status pins for parity error, framing error, and overrun. The MCS-8 interprets a carriage-return character (C/R) as a message or command terminator.

2. Process Name Loading

The prototype system permits receive process name loading into Ring Interface Process Name Memory (PNM) using the teletype. The RI matches received message destination codes with process names previously loaded into its PNM by its host. This procedure allows a host to have strict control over what messages it sees.

3. Command Language

The MCS-8 system supports a Command Language for the passing of commands from the TTY to the Ring Interface. The MCS-8 scans the first non-blank character from the TTY message for a '$' command character. TTY messages starting with a $ are always interpreted as commands and will not be transmitted over the ring. The character following the $ determines command type. Additional characters, if needed, are arguments to the command type. Arguments are separated from the command type with the '=' character. A list of current commands and their meanings follow:

Command	Meaning	Arguments
D	Destination name set	Process name
R	Remove receive process name	Process name
S	Source/Receive name set	Process name
A	Activate RI (join the ring)	
E	Exit the ring	

For example, typing '$S=C' followed by a carriage

return causes the MCS-8 to load the 8-bit (7-bit + even parity) USASCII code character "C" (43 hex) as a valid receive code into RI PNM. This same character code will be stored in MCS-8 RAM as the source field of outgoing messages. Up to 256 receive codes can be loaded into RI RAM, but only the most recently entered will be used as the source code on message transmission. Internally, when the MCS-8 receives the carriage return, it loads its RI output port with the character and outputs a logic '1' on the SET line to the RI. The RI uses the character as an address to its PNM and sets the bit determined by this address to a logic '1'. The RI acknowledges both the 's' and 'r' commands using the XMTR READY line.

A more general system would interpret the characters following the equal sign as a based number. Using the USASCII codes for process names limits valid names to those characters (less rubout and carriage-return) available on the ASR-33. This is no real limitation to the test and evaluation role of the MCS-8/TTY host.

IV. MCS-8 WITH TTY AS A HOST

A. HARDWARE CONFIGURATION

1. System Components

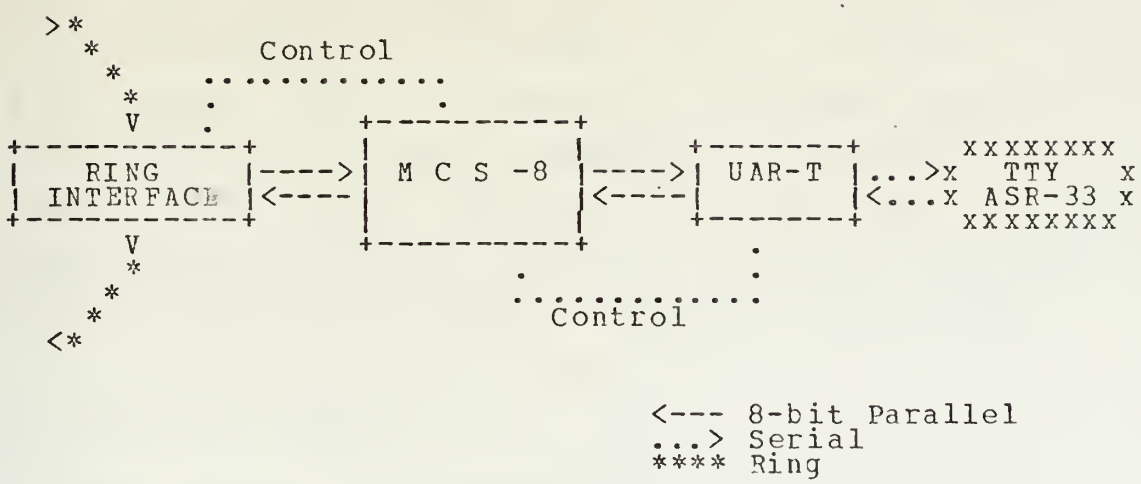


Fig.3. /mCS-8/TTY Host

Figure 3 shows the major components of the prototype MCS-8/TTY host. The micro-computer system is built up using INTELLEC-8 modules. The INTELLEC-8 Central Processor Module (8001-1 CPU), Input/Output Module, PROM Memory Module, and a custom 256X8 bit random access semiconductor memory (RAM) make up the MCS-8 system. The 8008-1 CPU executes non-memory referencing instructions in 12.5 microseconds when operated from the 800KHz clock contained on the CPU Module. The MCS-8 program requires 5 ROM's and 1 RAM.

The INTELLEC-8 I/O module contains the necessary TTY connections, the Universal Asynchronous Receiver-Transmitter, and the interface circuitry to the input and

output ports. The TTY is a common carrier TWX Terminal Model ASR-33 (8-level USASCII code, 110 bps).

2. Universal Asynchronous Receiver Transmitter (UAR-T)

The UAR-T is a general purpose, programmable MOS/LSI subsystem integrated on a single monolithic chip. It is used to convert asynchronous serial binary characters from the TTY to an 8-bit parallel format for input to the MCS-8.

The UAR-T also converts 8-bit parallel binary characters to serial, asynchronous output with start and stop bits added. Both the receiver and transmitter are double buffered. The I/O module provides automatic Receive Data Enable (RDE), and Reset Data Available (RDA) logic to the UAR-T when an input instruction is executed on the MCS-8. Data Strobe (DS) is automatic when an output instruction is executed. Automatic strobing is accomplished by MCS-8 instruction decoding within the I/O module.

B. SOFTWARE SYSTEM

The program for the MCS-8/TTY host is included as part of this thesis. It is programmed in PL/M [6]. Compilation into MCS-8 object code was acomplished on the IBM-360/67 using CP/CMS. Execution of the program was simulated using the FORTRAN IV program INTERP/8 [5]. This program simulates the 8008 CPU and provides monitoring commands to aid in program development. INTERP/8 accepts machine code produced by PASS 2 of the PL/M compiler, along with execution commands from the CP/CMS terminal. It also provides tracing features that allow CPU monitoring.

Procedure MAIN is the program executive routine. The basic loop consists of checking (1) the teletype for key depression and (2) the Ring Interface for message destination match with host process name (i.e., message for this host).

1. Program Flow

When a TTY key depression is detected (logic '1' on the 'DA' line from the UAR-T), the character is fetched from the UAR-T using Input port 0. The 'DA' line is reset automatically when the input instruction is executed, thereby re-enabling the UAR-T for the next character from the TTY. Messages from the TTY are accessed a character at a time and are terminated by the carriage return character. Messages are of two types, either command (first character a '$') or messages for transmission.

Port	Bit	Use
IN0	0-7	TTY data line from UAR-T
IN1	0	Character available from UAR-T
	1	Transmit Buffer Empty on UAR-T
	2	SOM/Next Byte Available from RI
	3	EOM from RI
	4	RVCR overrun/Match
	5	RVCR error (CRC)/Accept
	6	Transmitter (RI) Ready
	7	
IN2	0-7	Ring data line from RI
OUT1	0-7	Ring data line from MCS-8
OUT2	0	
	1	
	2	Activate/Exit
	3	Disable RI Receiver
	4	Byte Received from RI Byte Available to RI
	5	Enable RI transmitter
	6	Set host process name
	7	Delete host process name
OUT3	0-7	TTY data line from MCS-8

Fig.4. MCS-8 I/O Port Usage

Figure 4 lists the MCS-8 input and output port usage. When a command message is received, output port 1 is set with the process name, if required, and control signals are sent to the Ring Interface using output port 2. When a

message for transmission is received, the first byte is made available to Output port 1 and the RI is signalled using the XMTR ENABLE line. Procedure XMIT then completes message transfer to the RI.

The RI signals message received from another host using the SOM line. MAIN calls on procedure GETMSG to transfer and store bytes from the RI. The RI makes the determination of whether an error (overrun or CRC) has occured and sets the match, accept, and bad CRC bits accordingly. On an "accept" condition, procedure PRINTMSG is called to dump the message to the TTY.

2. File Codes

System flexibility is realized in the transmission of messages. Messages and message format are source process/destination process dependent. A possible application of the MCS-8/TTY host to the operational ring could be realized by the use of a message format using file codes. For example, with appropriate software support on the XDS-9300 or IBM-360, the MCS-8/TTY host can be used to create and execute files on a remote processor. The transmitted message would consist of a file code and an argument list. For example, 'NEW FILE1,NPS' could order the creation of a new file named 'FILE1' with a protect key of 'NPS'. After a file has been opened, file codes for "append to file," "delete current line," "change current line," etc., could be used for file creation and editing. File manipulation codes could also be implemented on the processors. "Erase file," "copy to printer," and "copy to TTY" are just a few.

With additional software support from a processor, the MCS-8/TTY host could be used in a desk calculator mode using appropriate calculator codes. Most calculator functions could be performed locally with message transmission only when required (such as for transcendental

functions).

V. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

A ring-switched computer network, using micro-computers at the host interface to the ring, can be designed to satisfy the requirements of reliability, flexibility, simplicity, and low cost. The major sacrifice in the prototype system is in the system bandwidth do to the relatively slow speed of the micro-computer, but the benefits in flexibility of design and system simplicity outweigh this deficiency. The DCS type organization with its distributed control philosophy is a reliable approach to network architecture. Difficulties in the DCS with regard to synchronizing and detecting loss of synchronization have been overcome in the NPS Ring architecture at a sacrifice in ring speed. The pseudo-bipolar violation concept provides the capability for variable length messages, ring synchronization, and control passing. The fact that these violations cannot appear in message data, forms a reliable basis for network protocol.

The MCS-8/TTY host provides a flexible access to the ring for system evaluation. The software provided will support a number of TTY modes, depending on the amount of interpretive support available on the major system processors.

B. RECOMMENDATIONS

Further work is required in several areas. It is recommended information security on the ring be investigated. The ring as it stands has no real protection on data flow. All data passes through each active node.

The present system does not provide for a software diagnostic routine to aid in fault isolation. Process names could be reserved for this purpose. Nucleus programs could communicate directly using these process names to, for example, selectively order an exit from the ring. Provision should be made to collect data on system performance. The average time it takes to get control of the ring for transmission, the number of messages requiring re-transmission, and the mean time between LOS conditions are considered important in determining system effectiveness.

```
 /*
          PASS1 COMPILER OPTIONS
*/
$MEM
$RIGHT=80
$WIDTH=120
/*
          GLOBAL VARIABLES
*/
DECLARE (MSGIN,MSGOUT) (80) BYTE;
DECLARE (INCOUNT,OUTCOUNT) BYTE;
DECLARE (OUT1,OUT2,OUT3) BYTE;
DECLARE (CHAR,CARRIAGE$RETURN,RUBOUT) BYTE
   INITIAL (0CDH,0ODH,0FFH);
DECLARE LIT LITERALLY 'LITERALLY', DCL LIT 'DECLARE';
DCL P0 LIT '0', P1 LIT '1', P2 LIT '2', P3 LIT '3';
DCL TRUE LIT '0FFH',FALSE LIT '0';
DCL (SERIAL,ACCEPTED) BYTE;
DCL (DCHAR,DESTINATION,ACTIVATE) BYTE;
DCL (EXIT,DELETE,SET,CHANGECK) BYTE;
DCL (COMMAND,EQUAL) BYTE INITIAL (FALSE,FALSE);
DCL LINE$FEED BYTE INITIAL (0AH);
/*
          G E T $ M S G
*/
/*
      GET MESSAGE FETCHES BYTES FROM THE RI
         UNTIL AN EOM IS DETECTED BY THE RI.
         THE PROCEDURE THEN WAITS FOR THE
         RI TO MAKE CRC AND OVERRUN STATUS
         BITS ABAILABLE.
      ON RETURN IF 'INCOUNT' IS GREATER
```

```
          THAN ZERO THEN THE MESSAGE RECEIVED
          IS ERROR-FREE AND CONTAINS 'INCOUNT'
          NUMBER OF BYTES.
*/
GET$MSG:  PROCEDURE;
/*
       RING INTERFACE HAS SIGNALLED MSG RECEIVED
          VIA INPUT PORT 1 BIT 2 SET AT LOGIC '1'
*/
/*
       INPUT PORT 2 HAS THE NEXT BYTE FROM THE RING
       INPUT PORT 1 BIT 3 AT LOGIC '1'
          INDICATES EOM DETECTED BY THE RI.
       INPUT PORT 1 BIT 2 AT LOGIC '1'
          INDICATES NEXT BYTE AVAILABLE.
       OUTPUT PORT 3 BIT 4 AT LOGIC '0'
          INDICATES BYTE STORED IN RAM.
       OUTPUT PORT 3 BIT 4 AT LOGIC '1'
          INDICATES MCS-8 HAS RECOGNIZED LOGIC '1'
          AT INPUT PORT 1 BIT 4 FOR NEXT BYTE AVAIL.
*/
/*
*/
    DCL J BYTE;
/*     STORE FIRST BYTE IN RAM  */
    MSGIN=INPUT(P2);
    J=1;
/*
       INFORM RING INTERFACE THAT BYTE HAS BEEN STORED
*/
    GMLOOP1:  OUT2,OUTPUT(P2)=OUT2 OR 10H;
/*        RETURN IF RING INTERFACE HAS RECEIVED AN EOM  */
       IF (INPUT(P1) AND 08H) > 0 THEN GO TO GET$XIT;
/*        WAIT UNTIL NEXT BYTE IS AVAILABLE  */
       GMLOOP2:
          IF (INPUT(P1) AND 10H) = 0 THEN GO TO GMLOOP2;
```

```
/*            NOTIFY RI BYTE AVAILABLE ACKNOWLEDGED  */
            OUT2,OUTPUT(P2) = OUT2 AND 0EFH;
            MSGIN(J)=INPUT(P2);
            J=J+1;
            GO TO GMLOOP1;
      GET$XIT:
/*
         CHECK RECEIVE STATUS FLAGS -- OVERRUN AND CRC
            INPUT PORT 1 BIT 4 AT LOGIC '1' INDICATES OVERRUN
            INPUT PORT 1 BIT 5 AT LOGIC '1'
               INDICATES CRC CHECK FAILED.
*/
         IF (INPUT(P1) AND 30H) > 0 THEN INCOUNT=0;
            ELSE  INCOUNT=J;
         RETURN;
END GET$MSG;
/*
            P R I N T $ C H A R
*/
/*
        PRINT CHARACTER OUTPUTS A CHARACTER
           IN USASCII CODE TO OUTPUT PORT
           P3.  NOTE THAT MCS-8 OUTPUT PORTS
           COMPLEMENT DATA.
*/
PRINT$CHAR:  PROCEDURE(PCHAR);
      DCL PCHAR BYTE;
/*
         WAIT FOR TRANSMITTER BUFFER REGISTER EMPTY ON UAR-T
         TBE='0' IMPLIES NOT EMPTY
*/
      PLOOP1:  IF (INPUT(P1) AND 02H)=0 THEN GO TO PLOOP1;
/*
         LOAD CHAR INTO OUTPUT PORT 0
*/
      OUTPUT(P3)=PCHAR XOR 0FFH;
```

```
/*
      NOTE:  UAR-T RESETS TBE TO LOGIC '1' WHEN TRANSMITTER
         REGISTER IS READY FOR NEXT CHAR
*/
    RETURN;
END PRINT$CHAR;
/*
         X M I T
*/
/*
      TRANSMIT (XMIT) MAKES MESSAGES FROM
         THE TTY AVAILABLE TO THE RI FOR
         TRANSMISSION.  AFTER TRANS. IS
         COMPLETE, IT WAITS AND CHECKS
         MATCH, ACCEPT, AND CRC BITS TO
         DETERMINE WHETHER THE MESSAGE
         HAS BEEN RECEIVED.
      MESSAGES NOT RECEIVED WILL BE RE-
         TRANSMITTED ONCE PRIOR TO AN
         ERROR '1' CONDITION.  ON RETURN
         'ACCEPT' IS TRUE IF TRANSMISSION
         WAS SUCCESSFUL.
*/
XMIT:  PROCEDURE;
    DCL XCOUNT BYTE;
/*
      MAKE DESTINATION AVAIL TO RI AT OUTPUT PORT 1
*/
    ACCEPTED=FALSE;
    XMIT$START:
      OUTPUT(P1)=MSGOUT;
/*
         NOTIFY RI MSG FOR TRANSMIT
*/
      OUT2,OUTPUT(P2)=OUT2 AND 0DFH;
/*
```

```
         CUTPUT ENTIRE MSG
*/
      DO XCOUNT=1 TO OUTCOUNT;
/*
            WAIT FOR RESPONSE FROM RI THAT
               LAST BYTE WAS RECEIVED.
*/
         XLOOP:IF (INPUT(P1) AND 40H) = 0 THEN GO TO XLOOP;
         OUT2,OUTPUT(P2)=OUT2 OR 10H;
         OUTPUT(P1)=MSGOUT(XCOUNT);
/*
            INFORM RI NEXT BYTE AVAILABLE
*/
         OUT2,OUTPUT(P2)=OUT2 AND 0EFH;
      END;
/*
      SERIAL BIT REQUIRED NEXT
      RESET MSG FOR TRANSMIT FLAG
         SIGNIFIES SERIAL BIT COMING.
*/
   XLOOP1:  IF (INPUT(P1) AND 40H) = 0 THEN GO TO XLOOP1;
   OUT2,OUTPUT(P2)=OUT2 OR 10H;
   OUTPUT(P1)=SERIAL;
/*
      WAIT FOR MATCH/ACCEPT BITS RECEIVED
*/
   XLOOP2:  IF (INPUT(P1) AND 40H) = 0 THEN GO TO XLOOP2;
/*
      IF MATCH OR ACCEPT BIT NOT SET RETRANSMIT MSG ONCE
*/
   IF (INPUT(P1) AND 30H) < 20H THEN
      IF (SERIAL = 0) THEN
         DO;
            SERIAL=1;
            GO TO XMIT$START;
         END;
```

```
        ELSE GO TO XOUT;
/*
        MESSAGE ACCEPTED
*/
    ACCEPTED=TRUE;
    XOUT:
        SERIAL=0;
        RETURN;
END XMIT;
/*
          P R I N T $ M S G
*/
/*
      PRINT MESSAGE OUTPUTS RECEIVED MESSAGES
        TO THE TTY VIA THE UAR-T.
*/
PRINT$MSG:  PROCEDURE;
    DCL FROM$MSG DATA ('FROM ');
    DCL PCOUNT BYTE;
/*
        OUTPUT CARRIAGE RETURN TO TTY
*/
    CALL PRINT$CHAR(CARRIAGE$RETURN);
/*
        OUTPUT 'FROM '
*/
    DO PCOUNT=0 TO 4;
        CALL PRINT$CHAR(FROMMSG(PCOUNT));
    END;
/*
        OUTPUT SOURCE NAME
*/
    CALL PRINTCHAR(MSGIN(1));
/*
        OUTPUT CARRIAGE RETURN
*/
```

```
    CALL PRINT$CHAR(CARRIAGERETURN);
/*
      OUTPUT MESSAGE
*/
    DO PCOUNT=2 TO INCOUNT;
      CALL PRINT$CHAR(MSGIN(PCOUNT));
    END;
    RETURN;
END PRINT$MSG;
/*
          E R R O R
*/
/*
     ERROR PRINTS 'ERROR' FOLLOWED
       BY THE ERROR NUMBER GIVEN AS
       THE CALLING ARGUMENT.
*/
ERROR:  PROCEDURE(ERRORNUMBER);
    DCL (ERRORNUMBER,ECOUNT) BYTE;
    DCL ERROR$MSG DATA ('ERROR ');
/*
      OUTPUT CARRIAGE RETURN
*/
    CALL PRINTCHAR(CARRIAGERETURN);
/*
      PRINT 'ERROR' FOLLOWED BY ERROR NUMBER AT TTY
*/
    DO ECOUNT=0 TO 5;
      CALL PRINTCHAR(ERRORMSG(ECOUNT));
    END;
    CALL PRINTCHAR(ERRORNUMBER);
    RETURN;
END ERROR;
/*
*/
/*
```

```
          M A I N   P R O G R A M
*/
/*
*/
    DCL I BYTE INITIAL (2);
/*
       CLEAR CUTPUT PORTS TO LOGIC '0'
       NOTE:  ALL CUTPUT PORTS COMPLEMENT DATA
*/
    CUT3,OUTPUT(P3),OUT2,OUTPUT(P2)=0FFH;
    RDA:
      IF (INPUT(P1) AND 01H) > 0 THEN
      /*
          FETCH CHAR FROM UAR T
      */
      DO;
        CHAR=INPUT(P0);
        IF CHAR <> CARRIAGE$RETURN THEN
          DO;
          IF CHAR <> RUB$OUT THEN
            DO;
            MSGCUT(I)=CHAR;
            IF I = 2 THEN
               DO;
               IF CHAR = ' ' THEN GO TO SKIP;
               IF NOT COMMAND THEN
                  DO;
                  IF CHAR='$' THEN COMMAND=TRUE;
                  ELSE I=I+1;
                  END;
               ELSE DO;
                  IF CHANGECK THEN
                    DO;
                    CALL ERROR ('2');
                    SET,DELETE,DESTINATION,ACTIVATE,EXIT,
                    COMMAND,EQUAL,CHANGEOK=FALSE;
```

```
            END;
          IF (SET OR DELETE) AND EQUAL THEN
            DO;
            OUTPUT(P1)=CHAR;
            IF SET THEN MSGOUT(1)=CHAR;
            CHANGEOK=TRUE;
            END;
          ELSE IF DESTINATION AND EQUAL THEN
            DO;
            DCHAR=CHAR;
            CHANGEOK=TRUE;
            END;
        ELSE IF ACTIVATE OR EXIT THEN
              CALL ERROR('2');
            ELSE IF CHAR='A' THEN ACTIVATE=TRUE;
              ELSE IF CHAR='E' THEN EXIT=TRUE;
            ELSE IF CHAR='S' THEN
              SET=TRUE;
              ELSE IF CHAR='D' THEN
                DESTINATION=TRUE;
                ELSE IF CHAR='R' THEN
                   DELETE=TRUE;
                   ELSE
                   IF CHAR= '=' THEN
                      EQUAL=TRUE;
                   ELSE
                   CALL ERROR('2');
          END;
        END;
      ELSE I=I+1;
      END;
    ELSE IF (I>2) THEN I=I-1;
    /*  CHAR IS RUBOUT AND I=2  */
      ELSE IF CHANGEOK THEN CHANGEOK=FALSE;
         ELSE
            SET,DELETE,DESTINATION,ACTIVATE,EXIT,
```

```
                COMMAND,EQUAL,CHANGECK=FALSE;
            END;
          ELSE IF (I>2) THEN
              DO;
                 OUTCOUNT=I;
                 CALL XMIT;
                 IF NOT ACCEPTED THEN
                    CALL ERROR('1');
                 OUTCOUNT=0;
                 I=2;
              END;
            /*  CHAR = C/R AND I=2  */
            ELSE DO;
              IF CHANGEOK THEN
                 DO;
                 IF SET THEN
                    OUT2,OUTPUT(P2)=OUT2 AND 08FH;
                 ELSE IF DELETE THEN
                    OUT2,OUTPUT(P2)=OUT2 AND 07FH;
                    ELSE MSGOUT=DCHAR;
                 END;
              ELSE IF ACTIVATE THEN
                 OUT2,OUTPUT(P2)=OUT2 AND 0FBH;
                 ELSE IF EXIT THEN
                    OUT2,OUTPUT(P2)=OUT2 OR 04H;
                    ELSE GO TO SKIP;
              SET,DELETE,DESTINATION,ACTIVATE,EXIT,
              COMMAND,EQUAL,CHANGEOK=FALSE;
/*
                 WAIT FOR ACKNOWLEDGEMENT FROM RI
*/
              PNAME: IF (INPUT(P1) AND 40H) = 0 THEN
                    GO TO PNAME;
              END;
        /*
            ECHO LAST CHARACTER TO TTY
```

```
      */
     SKIP: IF CHAR=CARRIAGE$RETURN THEN
              CALL PRINTCHAR(LINE$FEED);
           ELSE CALL PRINTCHAR(CHAR);
     END;
/*
      HAS RI RECOGNIZED MESSAGE FOR THIS PROCESS
*/
      IF (INPUT(P1) AND 04H) > 0 THEN
        DO;
          CALL GETMSG;
          IF ((INCOUNT>0) AND (CHAR=CARRIAGE$RETURN)) THEN
            DO;
               CALL PRINT$MSG;
               INCOUNT=0;
            END;
        END;
   GO TO RDA;
EOF
```

Proc. Symposium Computer Communications Networks and Teletraffic

1972

BIBLIOGRAPHY

1. Farber, D.J. and Larson, K.C., "The System Architecture of the Distributed Computer System--The Communication System," MRI International Symposium XXII on Computer Communication Networks and Teletraffic, Polytechnic Instutute of Brooklyn, April 1972.

2. Farber, D.J. and Larson, K.C., "The System Architecture of the Distributed Computer System--Software," MRI International Symposium XXII on Computer Communication Networks and Teletraffic, Polytechnic Instutute of Brooklyn, April 1972.

3. Farber, D.J. and Heinrich, F.R., "The Structure of a Distributed Computing System--The Distributed File System," Proceedings of the Conference on Computer Communications, p.364-370, October 1972.

4. Farmer, W.D. and Newhall, E.E., "An Experimental Distributed Switching System to Handle Bursty Computer Traffic," ACM Symposium on Problems in the Optimization of Data Communication Systems, Pine Mountain, Georgia, 13-16 October 1969.

5. INTEL Corporation, "8008 8-Bit Parallel Central Processor Unit Users Manual," MCS-8 Micro-Computer Set, November 1973.

6. INTEL Corporation, A Guide to PL/M Programming, September 1973.

7. Loomis, D.C., Ring Communications Protocols, UC Irvine Distributed Computer Project, Memo 46-A, May 1972.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Chairman Computer Science Group (Code 72) Naval Postgraduate School Monterey, California 93940	1
4.	Professor David J. Farber Department of Information and Computer Science University of California Irvine, California 92664	1
5.	Asst. Professor Ray Brubaker (Code 72Bh) Computer Science Group Naval Postgraduate School Monterey, California 93940	1
6.	LT Keith A. Hirt, USN Destroyer Development Group U. S. Naval Base Newport, Rhode Island 02840	1

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Prototype Ring-Structured Computer Network Using Micro-Computers		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis (December 1973) 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Keith Albert Hirt, LT, USN		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE December 1973 13. NUMBER OF PAGES 49
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Ring Network
Distributed Control
Computer Communication
Micro-Computer
PL/M
Computer Protocol
Network Synchronization

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

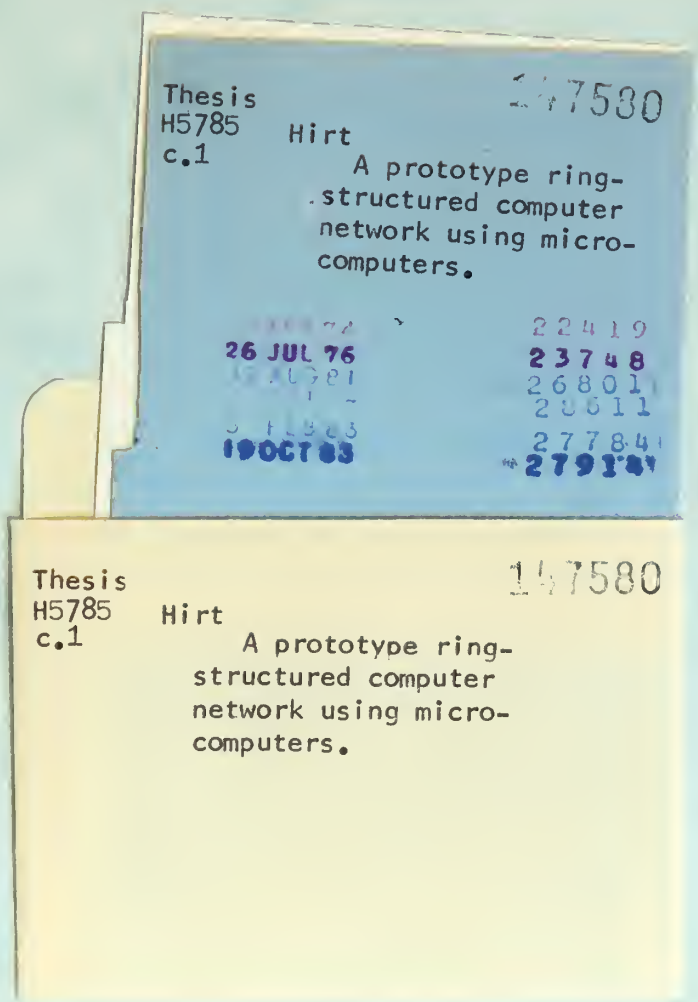
Architecture of a Ring communication network based on the Distributed Computing System (DCS) at the University of California, Irvine is presented. Control of the network is distributed in time among the active processors for reliability. A pseudo-bipolar violation protocol, modeled after the Farmer and Newhall ring at Bell Labs, is used to implement synchronization, control passing, and variable length messages. The prototype system is designed to make extensive use of micro-computers for flexibility in design.
(cont.)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |
(Page 1)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20.

Ring organization and protocol are discussed with the major emphasis on the use of an INTEL MCS-8 Micro-Computer at the interface of a teletype to the ring. The MCS-8 program is written in PL/M-a higher level language for INTEL's micro-computers.



Thesis
H5785
c.1

147580

Hirt
A prototype ring-structured computer network using micro-computers.